 **Pattern Databases Interaction**

**Course: Search in Artificial Intelligence**

**Problem: Tower of Hanoi with 4 pegs**

**Project Guide: Prof. Ariel Felner**

**By**

**Ran Weissman - 312236466**

**Uriel Zaed - 319435608**

**Background**

**Tower of Hanoi**

The Tower of Hanoi is a classic mathematical puzzle that has fascinated scholars since its invention by the French mathematician Édouard Lucas in 1883. The traditional version of the puzzle consists of three pegs and a few disks of different sizes, which can slide onto any peg. The puzzle starts with the disks neatly stacked in ascending order of size on one peg, the largest at the bottom, making a conical shape.

The objective of the game is to move the entire stack to another peg, obeying the following simple rules: only one disk can be moved at a time, each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty peg, and no disk may be placed on top of a smaller disk.

This creates a need for strategic planning and efficient problem-solving to complete the task in the minimum number of moves.

The classical Tower of Hanoi puzzle is a significant subject in computer science and mathematics, particularly in the study of algorithms and recursive functions. The minimum number of moves required to solve the puzzle is, where is the number of disks.

Extending the puzzle to four pegs introduces a new level of complexity. Known as the Reve's puzzle or the Tower of Hanoi with four pegs, this variant adds a layer of difficulty and requires more sophisticated algorithms for optimal solutions. The optimal solution for four pegs, unlike the three-peg version, does not have a simple closed-form formula.

**Pattern Database**

In the field of artificial intelligence and optimization, the use of pattern databases (PDBs) and heuristic search approaches is a powerful method for solving complex problems, such as the four-peg Tower of Hanoi puzzle. Pattern databases store precomputed solutions to subproblems, allowing an algorithm to quickly access these solutions during the search process, thereby reducing the computational effort required to find an optimal solution.

The pattern database heuristic approach begins by decomposing the original problem into smaller, more manageable subproblems. For instance, in the four-peg Tower of Hanoi, one might precompute solutions for various configurations of disks on three pegs and store these solutions in a pattern database. When solving the larger problem, the algorithm can then reference the PDB to quickly determine the best way to handle portions of the disks.

One popular search algorithm that benefits from pattern databases is A\*. A\* is a best-first search algorithm that uses a cost function, combining the actual cost to reach a node and the estimated cost to reach the goal from that node (the heuristic). In the context of the Tower of Hanoi, the cost function would incorporate the number of moves already made and the heuristic estimate from the PDB.

For example, suppose the heuristic function is based on the minimum moves required to solve a smaller subset of disks. During the search, A\* uses this heuristic to evaluate the states and prioritize the ones that seem closest to the goal. The use of a PDB ensures that these heuristic estimates are highly accurate, as they are derived from actual precomputed solutions, leading to more efficient search performance.

**Disjoint Additive Pattern Databases**

In solving a complex Towers of Hanoi problem, such as one involving 8 discs, pattern databases (PDBs) offer a practical approach to reduce computational complexity. A pattern database is a collection of precomputed solutions to smaller subproblems, which can be used to estimate the solution to the larger problem. For the 8-disc Tower of Hanoi, we can employ two separate PDBs: one for a 4-disc problem and another for a 4-disc problem.

Given any configuration of the 8 discs, they can be divided into two disjoint groups: one group containing 4 discs and the other containing 4 discs. By looking up the configurations of these two groups in their respective PDBs, we can sum the values obtained to get an admissible heuristic estimate for the 8-disc problem. This method ensures that the heuristic value is admissible because it does not overestimate the true minimal solution.

**PDB Heuristic – Suggestions**

1. **Pattern Databases Interaction (PDBI)**

Disjoint Additive Pattern Databases (DAPDBs) often fall short in accurately solving the entire problem of the Towers of Hanoi when dealing with large disc groups. This shortcoming arises because the heuristic estimate for the large disc group tends to be significantly low, failing to account for the movements required by the smaller disc group. The interaction between these groups, particularly the necessary movements of the smaller discs to accommodate the larger ones, is inadequately captured in this approach.

**Heuristic Stages:**

1. Pattern Database (PDB) for the Large Group:

2. Bucketing:

3. Augmented Bucketing:

4. Bucket Prices:

5. Generate Full Heuristic:

**Detailed Process:**

1. **PDB for the Large Group**: This initial step involves creating a PDB where each entry represents the minimal number of moves required for configurations consisting only of the large discs.

2. **Bucketing:** Group these configurations into buckets based on their PDB-determined costs. This bucketing helps in managing and simplifying the subsequent computations.

3. **Augmented Bucketing:** For each configuration in a bucket, consider the addition of the smaller disc group. These smaller discs are placed on pegs that are not utilized in the subsequent move indicated by the optimal solution from the PDB. This step integrates the dynamics of both disc groups, providing a more comprehensive view of the required moves.

4. **Bucket Prices:** Within each bucket, solve for the optimal solution cost for the whole problem. This involves finding the minimal number of moves needed starting from any configuration within the bucket, effectively capturing the interaction between the large and small disc groups.

5. **Generate Full Heuristic:** Finally, for any given configuration of the full problem, identify the cost from the PDB for the large group. Then, enhance this cost by incorporating the new heuristic value derived from the Bucket Prices. This combined approach yields a more accurate heuristic estimate that better reflects the true complexity of the problem.

This method aims to provide a more precise heuristic by addressing the limitations of the DAPDBs, particularly the underestimation due to the lack of consideration for the smaller disc group's movements. Through a more nuanced integration of both disc groups, the suggested heuristic seeks to deliver improved problem-solving performance for the Towers of Hanoi with large disc sets.

1. **Additive Pattern Databases Interaction (APDBI)**

This heuristic closely resembles the Pattern Databases Interaction (PDBI) with notable differences in stages 4 and 5.

**Stage 4:** Instead of solving the entire problem, we solve the subproblem involving only the large disc group. This focused approach aims to optimize the configuration and movements of the larger discs without considering the smaller discs initially.

**Stage 5:** The final heuristic value is derived differently. As in PDBI, the cost is initially determined using the bucket pricing method. However, this cost is then augmented by adding the value from the Pattern Database (PDB) specifically for the small disc group. This adjustment ensures that the movements and interactions of the smaller discs are accounted for in the final heuristic value, providing a more accurate and comprehensive estimate of the solution cost.

**Experiments**

Code in [Git](https://github.com/Uriel-Zaed/AI-Search-Project)

To evaluate our proposed heuristic, we implemented the Towers of Hanoi problem using Python. Our experimental setup focuses on the Hanoi problem with **four pegs and eight discs**. We divided the discs into two groups: a large group consisting of four discs and a smaller group also consisting of four discs.

For the large group, we created a Pattern Database (PDB) that captures the minimal number of moves required to solve configurations involving only these four discs. This PDB was generated using a Breadth-First Search (BFS) from the goal backwards. The PDB serves as the foundation for our heuristic calculations, providing precomputed optimal solutions for various configurations of the large disc group.

We then utilized an A\* algorithm, enhanced with the Additive PDB heuristic, to implement the Bucket Prices approach. This involved organizing configurations of the large group into buckets based on their PDB-determined costs. For each configuration within these buckets, we incorporated the smaller disc group by considering all possible placements on the available pegs.

We also examined the heuristic's admissibility by comparing it to an optimal solution calculated using a Breadth-First Search (BFS) from the goal state backward. This comparison ensures that our heuristic does not overestimate the true minimal solution cost, thereby confirming its accuracy and reliability in guiding the search algorithm.

Finally, we compared our proposed heuristics with the additive Pattern Database (PDB) heuristic by analyzing the number of expanded nodes in the A\* algorithm. This comparison serves to evaluate the efficiency and effectiveness of our heuristic approach in guiding the search process towards finding optimal solutions for the Towers of Hanoi problem with four pegs and eight discs.

**Results and Conclusion**

Upon confirming admissibility, it was established that the Pattern Databases Interaction (PDBI) maintains admissibility, whereas the Additive Pattern Databases Interaction (APDBI) does not. Consequently, our analysis concentrated on comparing PDBI with the Additive Pattern Database (APDB).

The comparative findings highlighted a significant improvement in node expansion efficiency. Specifically, APDB expanded across 27,953 nodes, whereas PDBI expanded across only 7,198 nodes. This substantial difference underscores the superior performance of PDBI in guiding the search algorithm towards optimal solutions for the Towers of Hanoi problem with four pegs. Importantly, these results open new possibilities for extending the heuristic approach to effectively solve problems involving more than 31 discs optimally.

**Project extension**

We apply the 4-peg Tower of Hanoi problem to configurations with 6 and 8 discs. We explore all scenarios by analyzing different groupings of discs (e.g., 7-1, 6-2, 5-3, etc.). For each scenario, we validate the admissibility of the PDBI heuristic.   
Additionally, we compare the effectiveness of the PDBI heuristic against the additive heuristic.

We conducted additional work on the bucket pricing stage (stage 4) of the PDBI heuristic calculation. This stage is the most time-consuming, so we aimed to gain a deeper understanding of its details.   
We discovered that the primary issue was the large number of initial states for each A\* search, leading to the expansion of numerous unnecessary nodes while searching for a solution. To address this, we analyzed each bucket within the bucket pricing stage more closely to better understand the underlying processes and optimize performance.

We took the 8 discs and ran the bucket pricing stage using groupings of 4-4 and 6-2.   
For each bucket in this stage, we collected data on the number of initial states, the number of nodes expanded, the minimal initial state (the one derived from the solution to the A\* search), and the number of nodes expanded for A\* when considering only the minimal initial state.

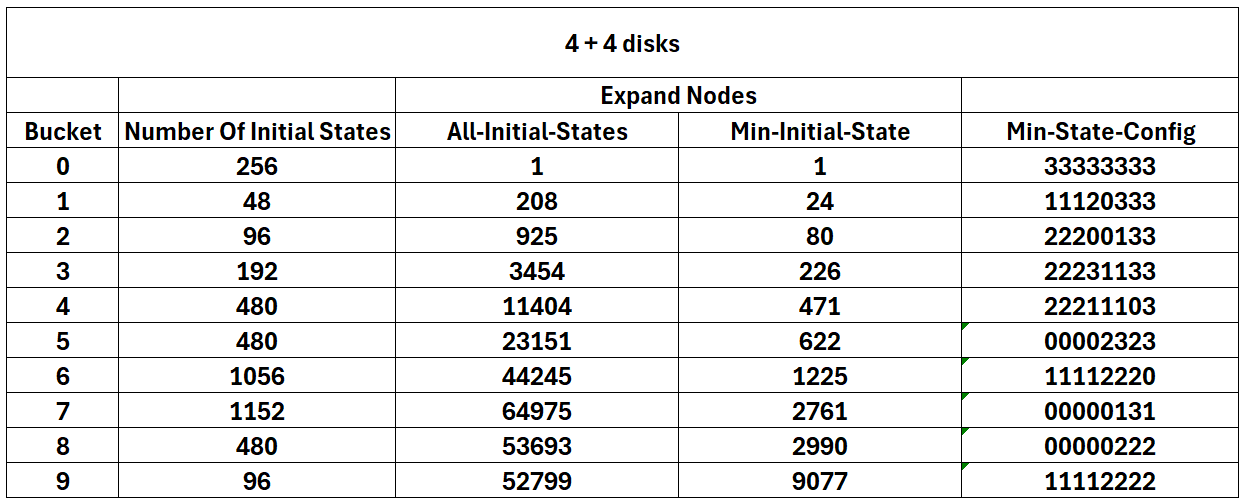
**Project extension - Results**

Comparison between the PDBI Heuristic and the Additive Heuristic on expansion nodes:

|  |  |  |
| --- | --- | --- |
| **6 disks** | | |
|
|  | **PDBI** | **Additive** |
| **1,5** | **1527** | **521** |
| **2,4** | **548** | **771** |
| **3,3** | **482** | **1032** |
| **4,2** | **271** | **787** |
| **5,1** | **86** | **374** |

|  |  |  |
| --- | --- | --- |
| **8 disks** | | |
|
|  | **PDBI** | **Additive** |
| **1,7** | **36638** | **8438** |
| **2,6** | **11539** | **15349** |
| **3,5** | **8777** | **23917** |
| **4,4** | **7198** | **27953** |
| **5,3** | **9564** | **25162** |
| **6,2** | **5559** | **15814** |
| **7,1** | **1561** | **5242** |

Detailed Analysis of Node Expansion and Initial States in the Bucket Pricing Stage of PDBI Heuristic Calculation:

****

**A table with numbers and a number

Description automatically generated**

**Project extension - Conclusions**

Our findings reveal that the admissibility of the PDBI heuristic remains unaffected by the grouping of discs. This is a significant observation, as it indicates the robustness of the PDBI heuristic across different problem configurations. However, the effectiveness of the PDBI heuristic varies with the size of the group.

Specifically, when using a large group of only one disc, the PDBI heuristic tends to perform poorly in comparison to the Additive heuristic. This is likely due to the limited scope of subproblems being addressed, which can lead to less accurate estimations of the overall solution cost.

Conversely, for larger groups beginning with two discs and continuing with more configurations, the PDBI heuristic demonstrates a notable advantage over the Additive heuristic.

One disadvantage of the PDBI method is the substantial time required to compute the A\* search for each bucket in the bucket pricing stage (stage 4), due to the large number of node expansions. The total number of expansions was considerably higher compared to the node expansions observed in BFS, even when considering the total number of configurations.

In summary, while the Additive heuristic may be more effective for simpler cases with smaller groups, the PDBI heuristic proves superior for larger and more complex configurations. This is attributed to its ability to leverage detailed precomputed information, leading to better-informed decisions during the search process.

**Future research**

As we mentioned, the primary challenge with this heuristic calculation is the time consumption in the bucket pricing stage (stage 4). Each bucket contains many initial states. We have observed that limiting the A\* search to only the minimal initial state significantly reduces the number of node expansions. This suggests that by reducing the number of initial states in each bucket, we could greatly improve the time complexity.

By thoroughly analyzing the minimal state configurations and examining the patterns in small groups, we can identify recurring patterns across various groupings and numbers of discs. Observing these patterns in numerous minimal state configurations allows us to find commonalities. Once a commonality is identified, we can reduce the number of augmented configurations in the Augmented Bucketing stage (stage 3), leading to a significantly smaller number of initial states in each bucket.

Top of Form

Bottom of Form